Virtual Rack®

State Service Manual

Part Number
VRSM-S1-750-DOC

Innovative Technologies, Inc.

Copyright © 2006 Innovative Technologies, Inc
All Rights Reserved

List of Figures:

VR State Service

The Virtual Rack® State service is used to persist the attributes and\or values of VR objects, recursively from any point in the system hierarchy. The State service is an invaluable tool for preserving hardware configuration settings, and reliably restoring those settings each time the system is started. The VR State service provides many advanced features for controlling how much state information is stored\loaded, merging additional state into an existing file, and controlling how objects are referenced within the file.

VR State Interface

f(x)	LoadState	Load a state XML file.
f(x)	StoreState	Store a state XML file. Requires a starting URL.

Figure 1 VR State Interface

VR State Context Menu

The VR State service adds the ability to store or load state on any VR Object.

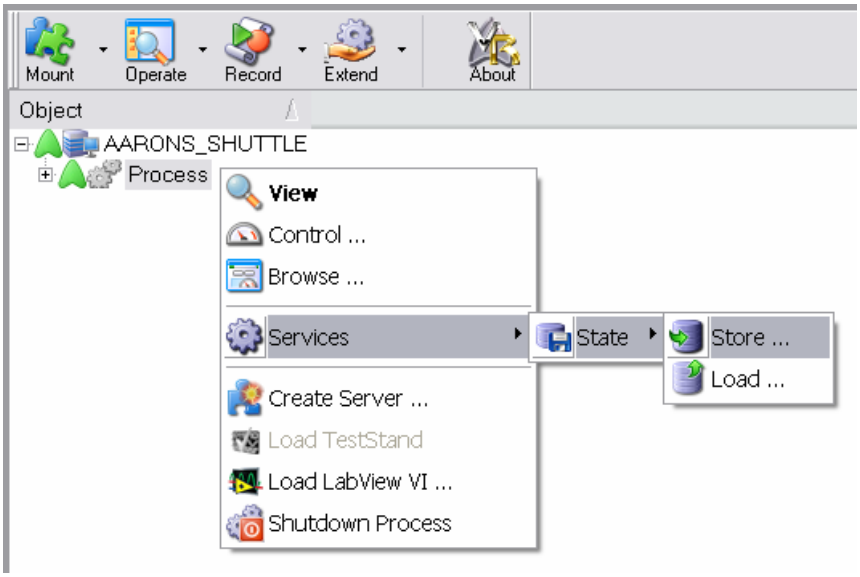

Figure 2 VR State context menu

Selecting either the state store or load command will query the user for a file path. Selecting a path to an existing file will merge the new state information into the file's existing state.

As an alternative to state merging, users can store state on multiple objects at once.

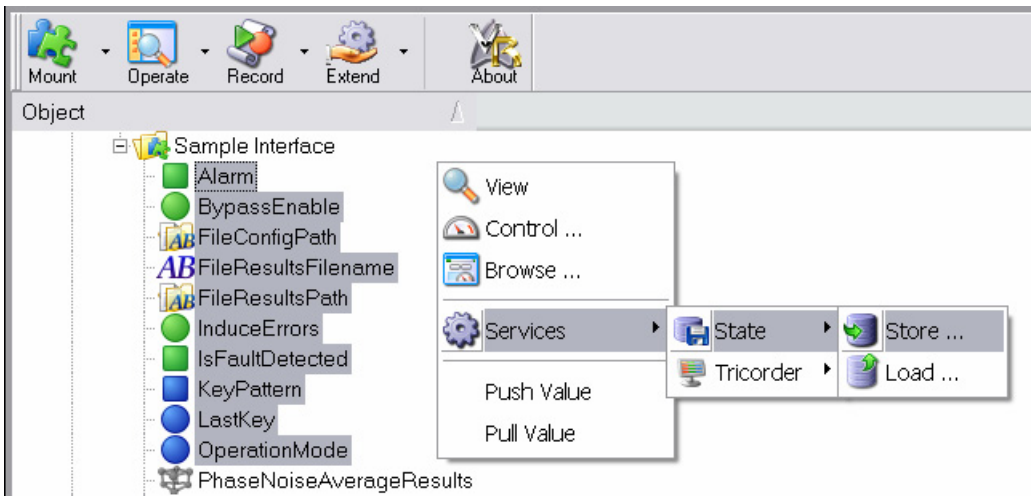

Figure 3 Multi-select state store

When loading state, the menu context only applies for state files that use relative path URLs.

VR State Tasks

The VR State service adds a load and store task to the task list of all VR objects.

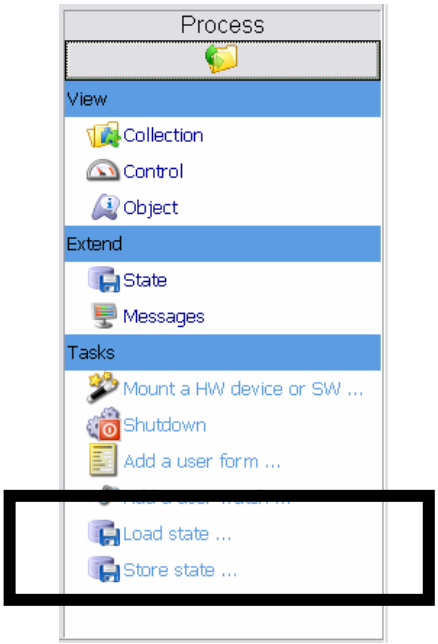

Figure 4 VR State task entries

VR State Views

The VR State view is a one stop shop for all things state. The context menu and state tasks only expose the most common state tasks. Advanced features can only be accessed from the VR State view.

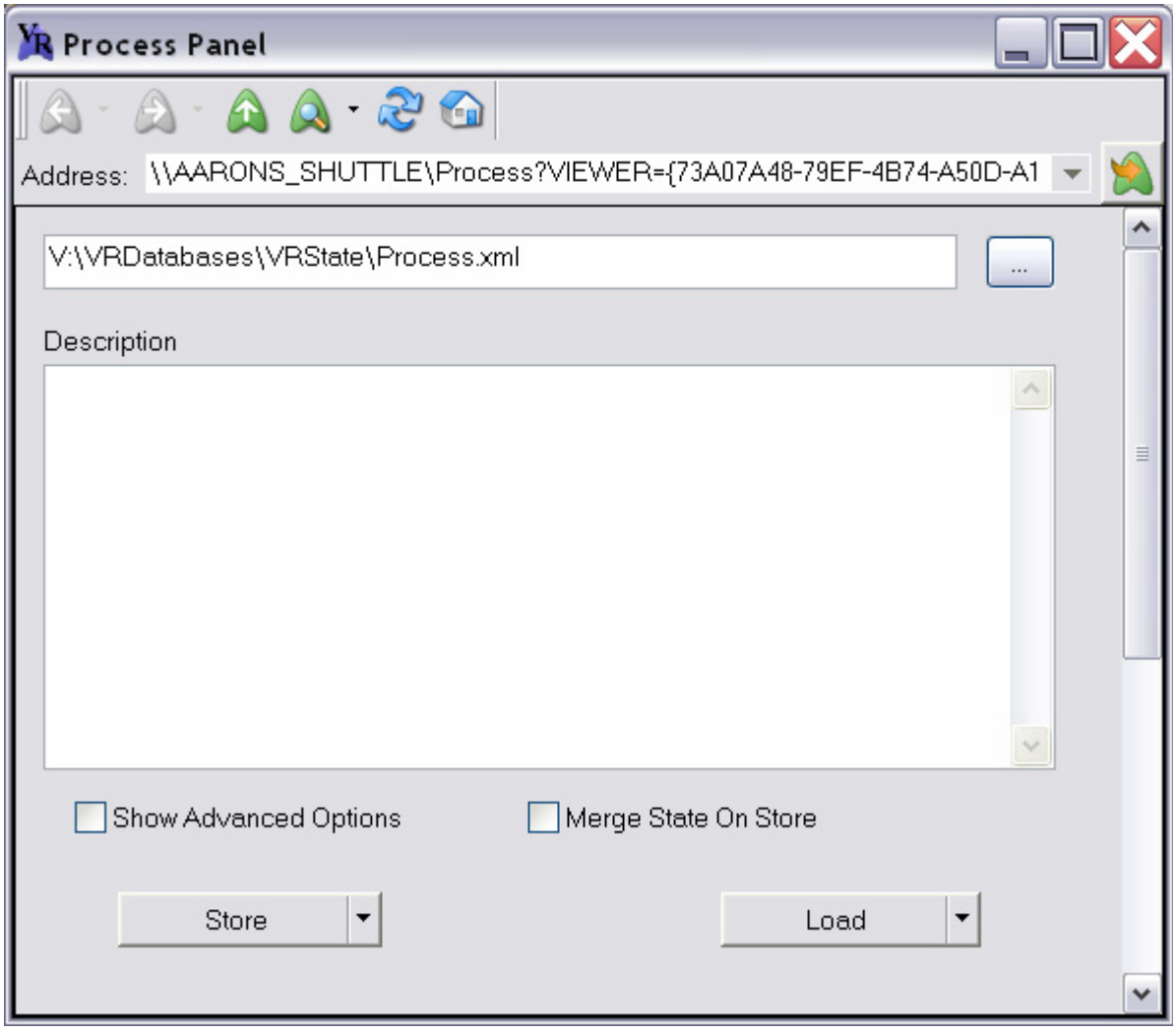

Figure 5 VR State view

The file path edit control at the top specifies the state file path used to store\restore state. Use the ellipsis button to browse to a different file.

Description

The description area allows the user to add\read notes about the state file. For example, after successfully configuring a Symmetricom BC635 card to synchronize with an incoming IRIG signal, the user may want to store the

setup configuration and leave notes about how the state file will affect the card configuration.

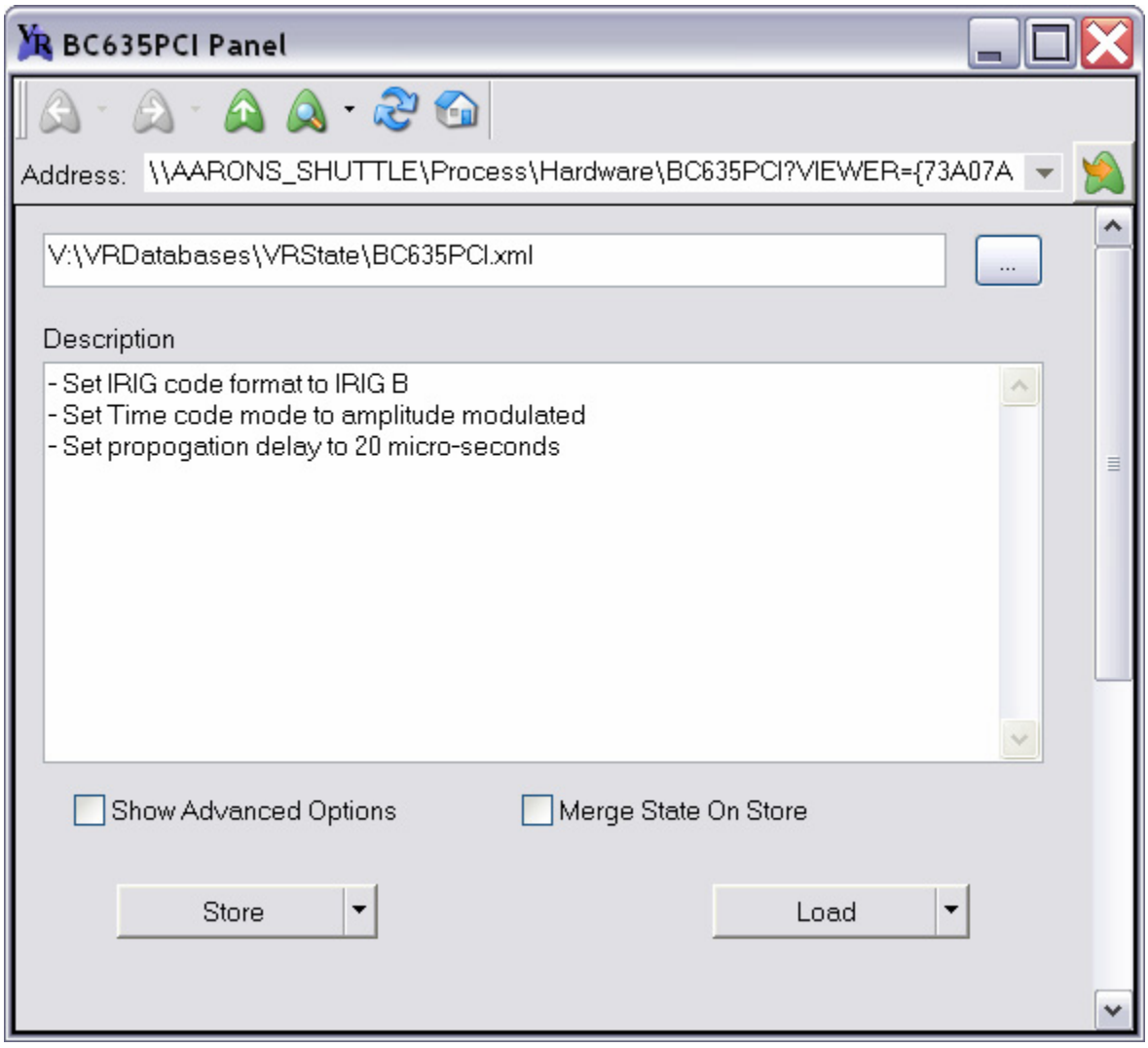

Figure 6 VR State file description example

These notes are persisted in the state file and become an important reference on racks with complex state files.

Merge State On Store

Use the 'Merge State On Store' check box to add additional state to an existing state file. Storing state to an existing file with 'Merge State On Store' unchecked will simply overwrite the file.

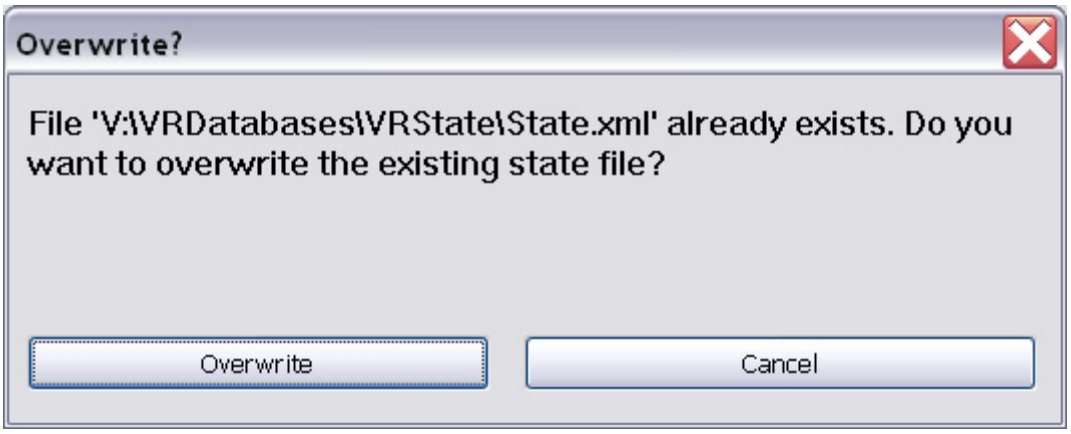

Figure 7 VR State store overwrite prompt

Click 'Overwrite' button to replace the state file. However, with the 'Merge State On Store' checked, storing to an existing file will prompt you to merge with the file's existing state information. This is the default behavior for the VR State context menu and tasks.

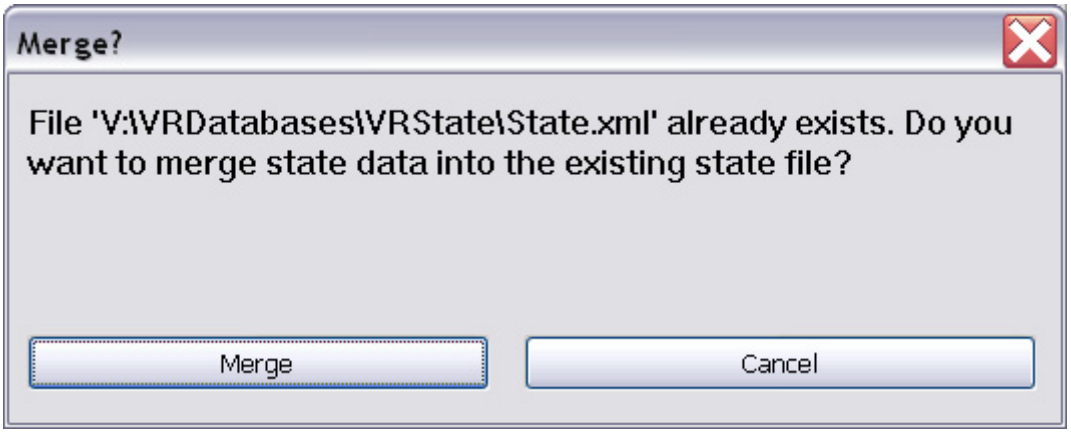

Figure 8 VR State store merge prompt

This feature is invaluable when users want to keep state in one file, but build that state one server at a time.

Recording

Recording of TestStand state store and load steps, use the drop down menus from the store and load buttons.

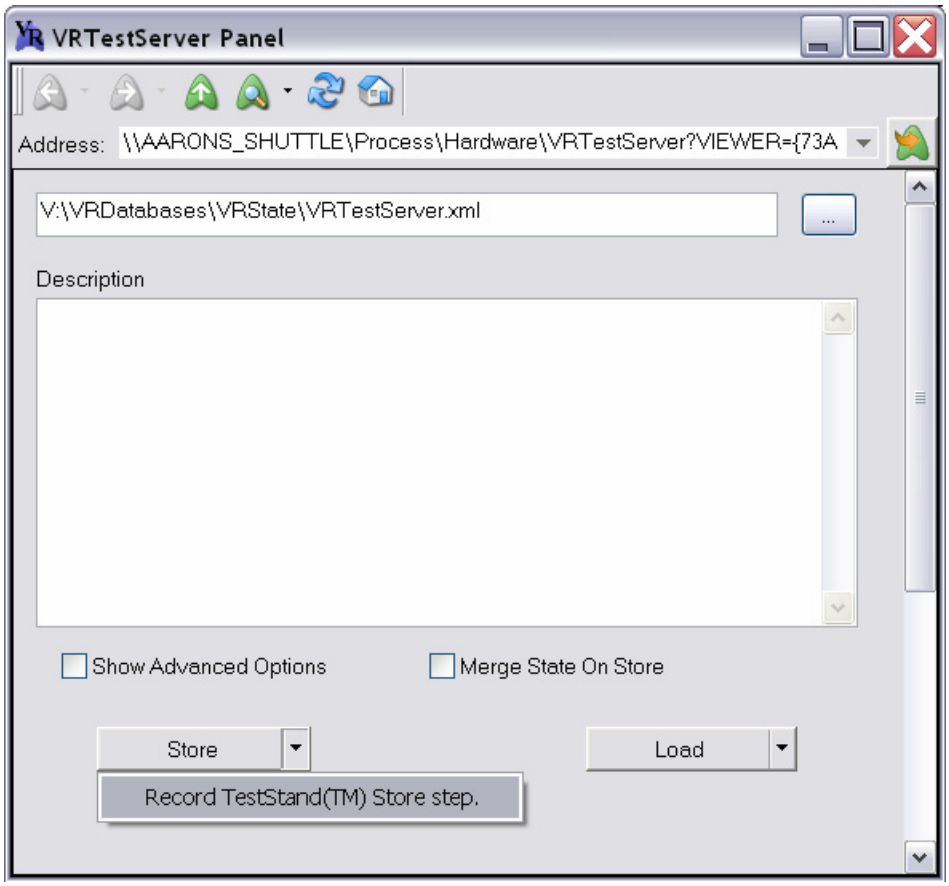

Figure 9 Recording a TestStand step

Recording a state step will start an active recording if it has not already been started. See the TestStand service chapter for further details.

Advanced Options

Use the 'Show Advanced Options' check box when finer control of the state file contents is required. Advanced state options should only be used by expert Virtual Rack® users.

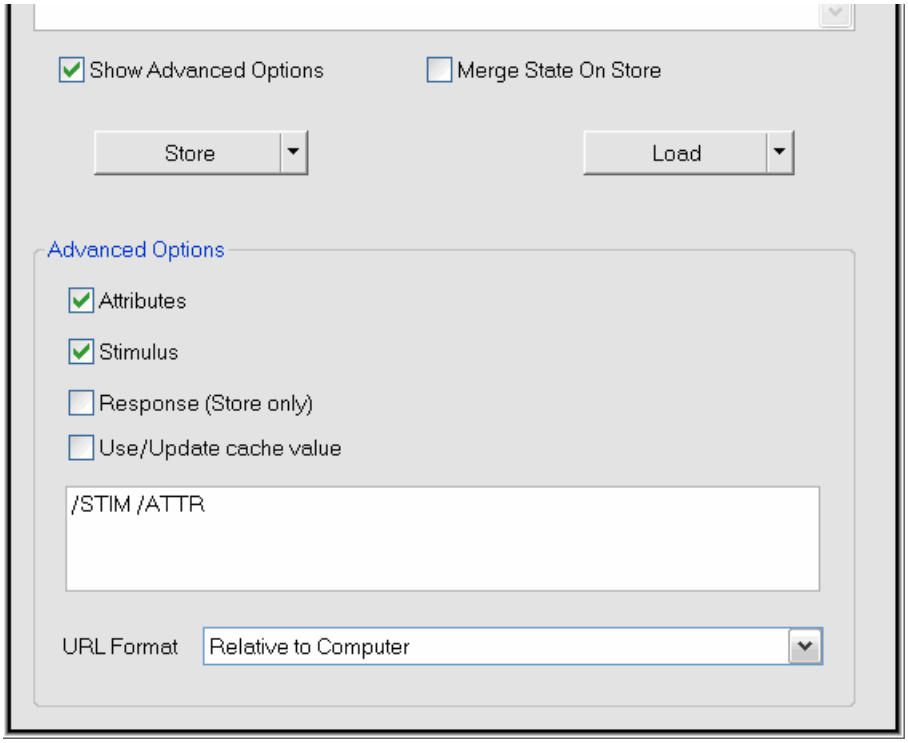

Figure 10 VR State advanced options

Attributes	Store the attributes of each object in the tree. Examples of object attributes are: label, units, display range, etc. This setting defaults to true, but users may want to disable attribute storage to reduce the size of the resulting state file, and the related load time of that state file.
Stimulus	Store the value of data type objects. This setting does not apply to non-data types, such as VR actions, VR collections, etc.
Response	Store the value of read only value types. Read only values will not be restored when the state file is loaded.
Use/Update cache only	Controls whether state loading and storage uses the cached property values. This setting is useful when storing state on a system with long hardware

access times. To reduce state storage times, store state using the cached values to avoid invoking a hardware access.

Use the URL format field to control how URLs are stored in the state file. The URL format affects how easily the state file can be re-used on different computers, or different object instances.

Absolute	URL is stored with the specific computer and instance name, making the state file compatible only with the computer and instance name it was created with. For example, storing state on the 'MyServer2' server will store state URLs as \\MyComputer\Process\Software\MyServer2. The state file will always target MyServer2 on MyComputer, regardless of what computer it is loaded from. This setting is used on enterprise systems where users want to load system state from any computer on the network.
Relative to Computer	URL is stored relative to the computer (eg. \\.\Process\Software\MyServer2). Loading the state file will always attempt to load to the servers running on the local computer. This configuration is the default, and desirable on single computer systems. Relative state files will continue to function when the computer name is changed, or if moved to a second workstation system with identical server names.
Relative to Process	URL is stored relative to the computer and process (eg. \\.\.\Software\MyServer2). This allows the user to move a server to a different process without invalidating state files.
Relative to Object	URL is stored relative to the parent object. Using this setting, the user can load the same state file into multiple instances of a server.

